Construction of Analytic Frameworks for Component-Based Architectures

George Edwards
gedwards@usc.edu

Chiyoung Seo
cseo@usc.edu

Nenad Medvidovic
neno@usc.edu

Computer Science Department
University of Southern California
Los Angeles, USA

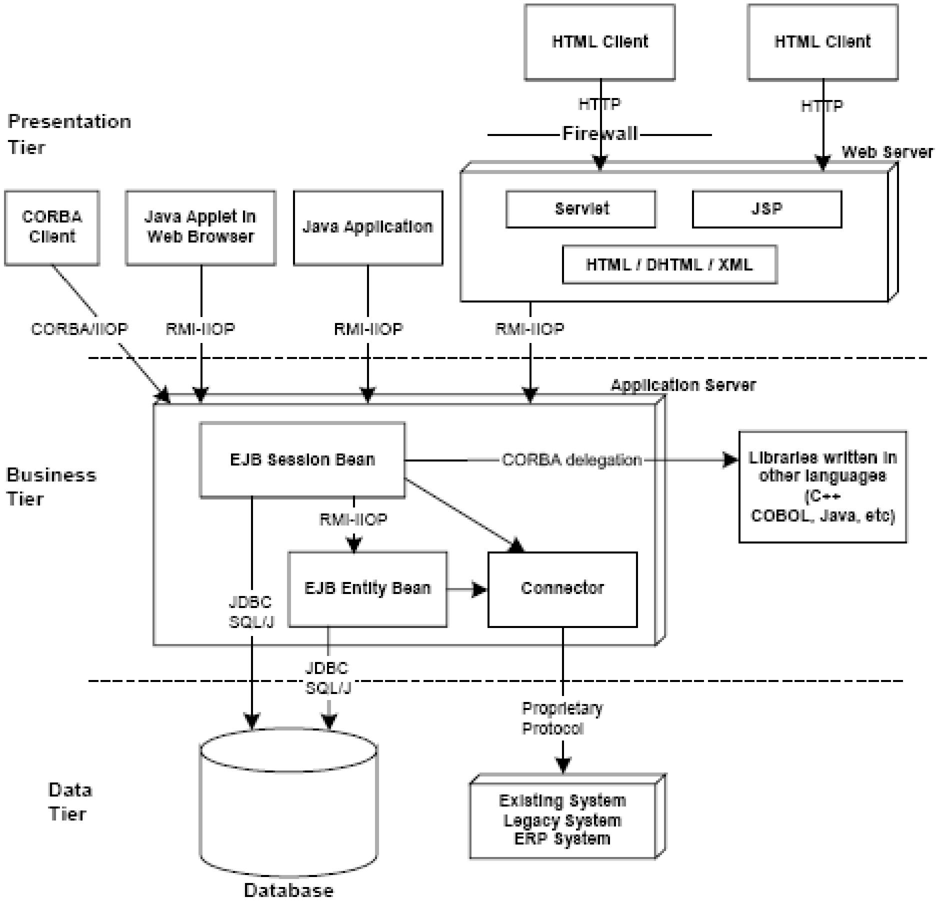
Presentation Outline

- Background and Motivation
- Model Interpreter Frameworks
  - Definition
  - Assumptions
  - Requirements
- The eXtensible Toolchain for Evaluation of Architectural Models (XTEAM)
  - Design
  - Evaluation
- Conclusions

Component Technologies

- ***Component technologies*** provide the basis for modeling, implementation, and deployment of software architectures
  - A ***component model*** defines the well-formedness of component instances/assemblies
  - A development platform/run-time environment enforces the component model

- Examples include Java EE, CORBA Component Model and the .NET Framework

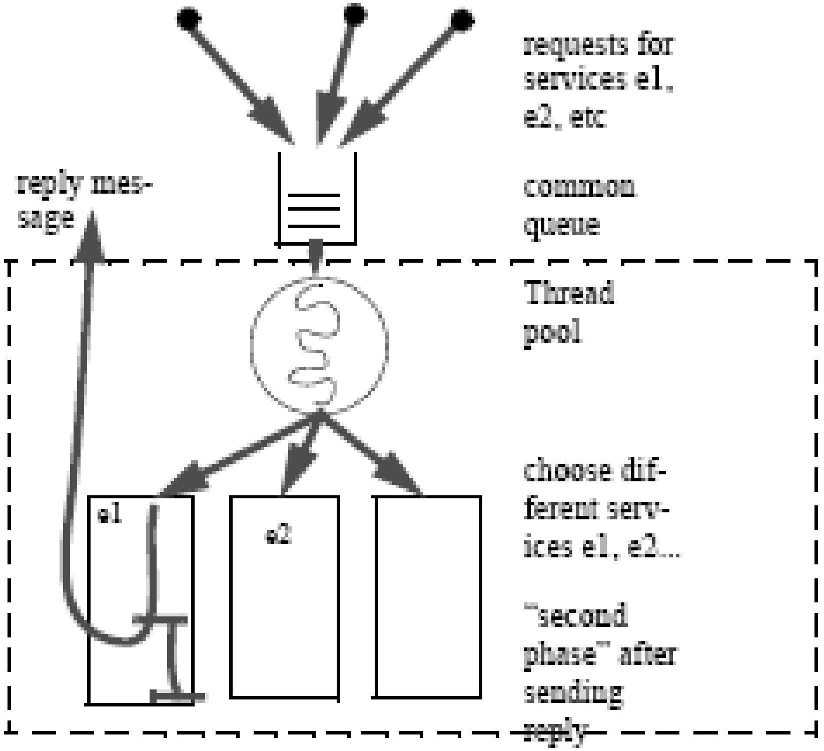


Java EE Component Model (informal)

Analysis Technologies

- ***Analysis technologies*** enable the prediction of the non-functional properties of software systems
  - An ***analysis technique*** defines a process for applying a computational theory to system models
  - Analysis techniques rely on specific assumptions about the systems to which they are applied

- Prediction of the non-functional properties of component-based architectures requires the integration of component technologies and analysis technologies


requests for services e1, e2, etc
reply message
common queue
Thread pool
e1
e2
choose different services e1, e2...
"second phase" after sending reply

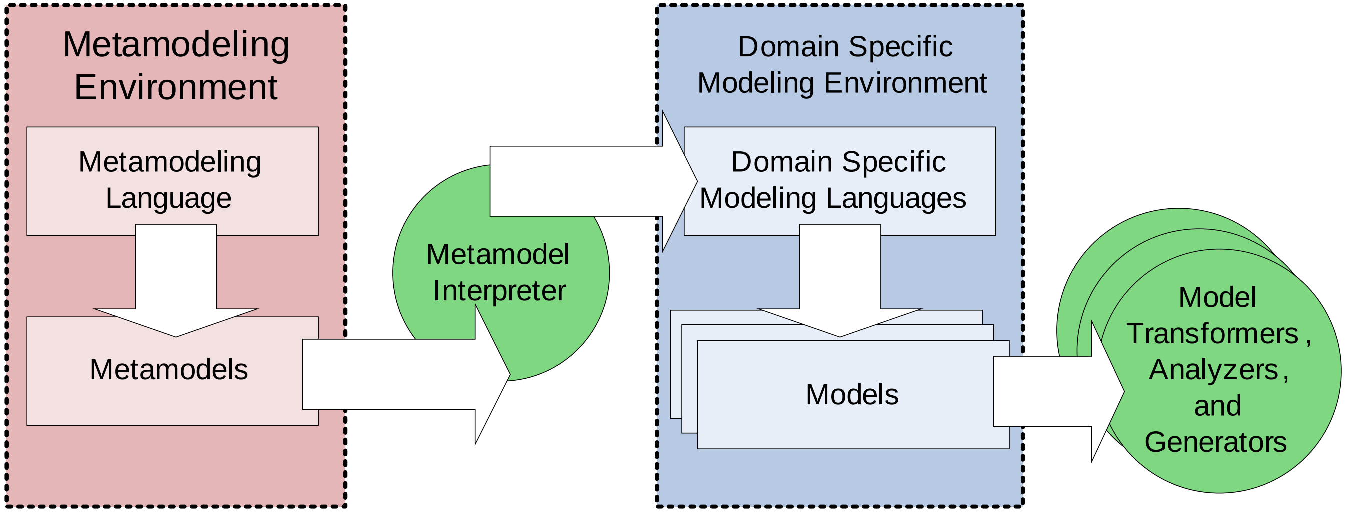
Layered Queuing Network (LQN) Analysis Model (informal)

Model-Driven Engineering

- ***Model-driven engineering*** (MDE) combines domain-specific modeling languages with model analyzers, transformers, and generators
  - Models are the central engineering artifacts throughout the engineering lifecycle
  - Domain concepts are codified as first-class modeling elements
  - Model transformations allow a single system model to be used for a variety of purposes

Metamodels define elements, relationships, views, and constraints

Model interpreters leverage domain-specific models for analysis, generation, and transformation


Metamodeling Environment
Metamodeling Language
Metamodels
Metamodel Interpreter
Domain Specific Modeling Environment
Domain Specific Modeling Languages
Models
Model Transformers, Analyzers, and Generators

Motivation for This Work

- **Problem:** Using the standard MDE process, a model interpreter must be constructed for *each* analysis that will be applied to a design model
  - Requires system architects and developers to become tool *developers* – rather than merely tool *users* – to achieve integrated design analysis
  - Organizations want to develop with third-party, commercially-supported tools to reduce risk and cost

- **Solution:** Use a model interpreter framework to implement architectural analyses
  - Allows tasks to be performed only *once* for a broad *family* of analysis techniques
  - Provides *built-in* analysis capabilities along with metamodeling and domain specific extensibility

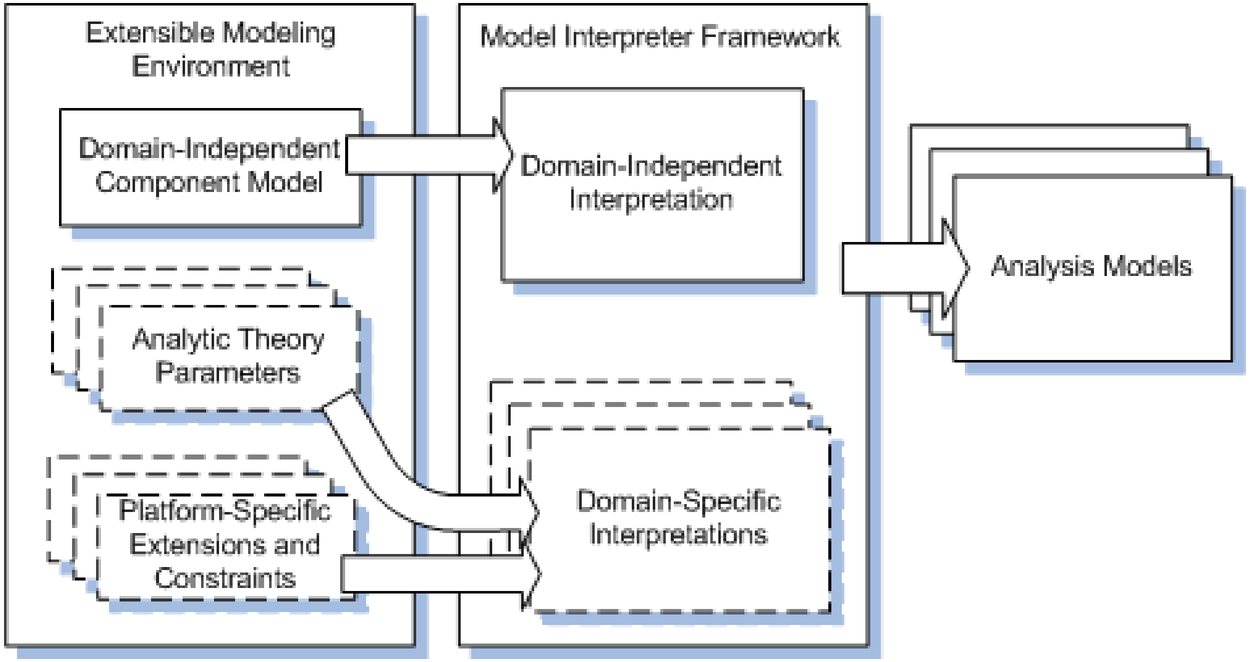
Model Interpreter Implementation Tasks

- Find a computational theory that derives the relevant properties
- Determine the syntax and semantics of the analysis modeling constructs
- Discover the semantic relationships between the constructs present in the architectural models and those present in the analysis models
- Determine the compatibility between the assumptions and constraints of the architectural models and the analysis models, and resolve conflicts
- Implement a model interpreter that executes a sequence of operations to transform an architectural model into an analysis model
- Verify the correctness of the transformation

Model Interpreter Frameworks

- An infrastructure for constructing a family of model interpreters

- Implement a semantic mapping between a component model and an analysis model


Extensible Modeling Environment
Domain-Independent Component Model
Analytic Theory Parameters
Platform-Specific Extensions and Constraints
Model Interpreter Framework
Domain-Independent Interpretation
Domain-Specific Interpretations
Analysis Models

- Provide extension mechanisms to accommodate domain-specific modeling and analysis

- Enable a family of analytic techniques to be applied to a component model

- Can be reused by a software architect to rapidly construct analysis models from domain-specific architectures

MIFs: Assumptions

1. System models contain domain-independent elements that are sufficient to implement an interpretation

3. The interpretation of domain-independent elements is not dependent on the interpretation of domain-specific elements

5. Domain-specific constraints do not violate domain-independent constraints

MIFs: Requirements

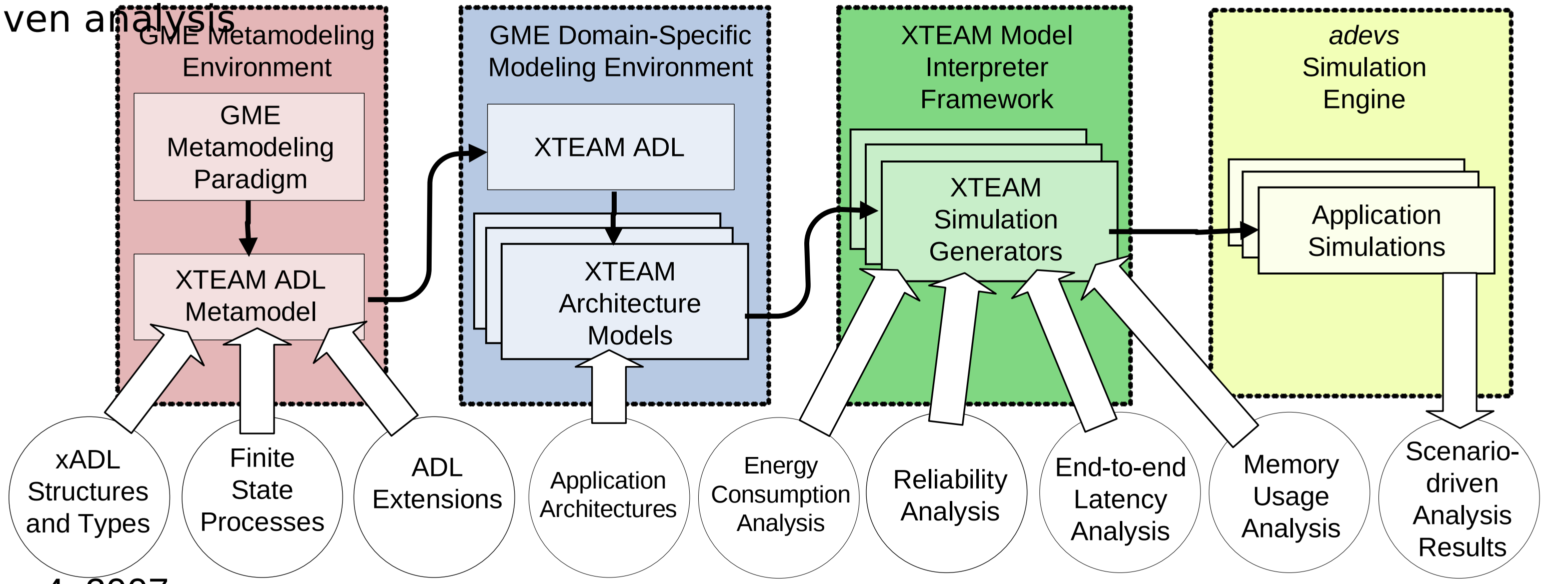
1. The model interpreter framework abstracts the details of domain-independent interpretation

3. The model interpreter framework produces an artifact useful in a wide variety of contexts

5. The model interpreter framework provides extension mechanisms sufficient to accommodate domain-specific interpretation

eXtensible Toolchain for Evaluation of Architectural Models (XTEAM)

- XTEAM employs a metaprogrammable graphical modeling environment (GME)
- XTEAM composes existing general-purpose ADLs: xADL Core (structures and types) and FSP (behavior)
- GME configures a domain-specific modeling environment with the XTEAM ADL
- Architecture models that conform to the XTEAM ADL are created
- XTEAM implements a model interpreter framework
- The XTEAM ADL is enhanced to capture domain- and platform-specific information
- The XTEAM Model Interpreter Framework is utilized to implement simulation generators
- Application simulations execute in the *adevs* discrete event simulation engine
- Simulations operate on the information captured in ADL extensions to perform scenario-driven analysis

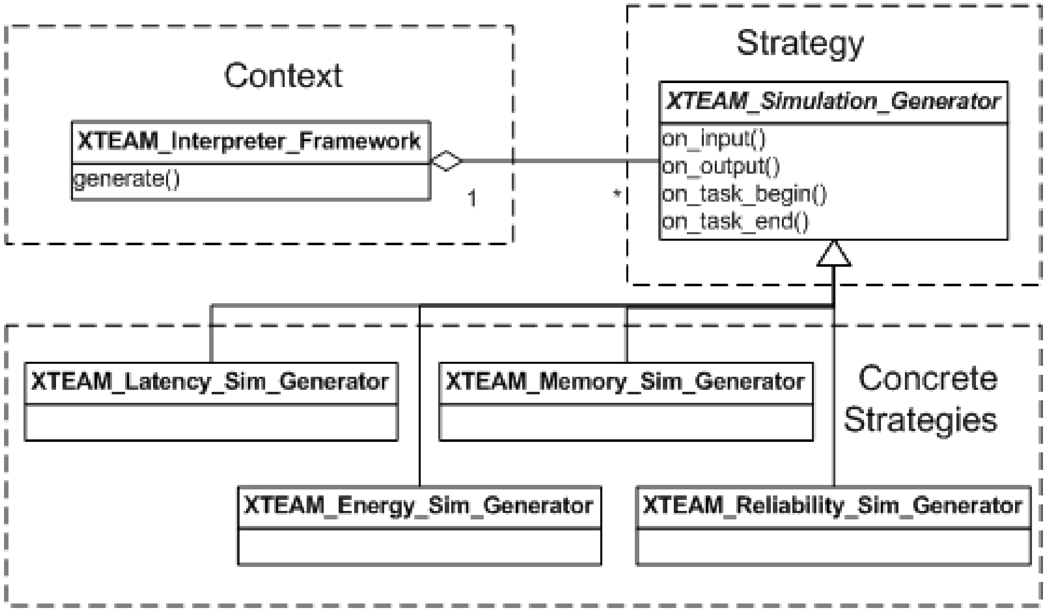

GME Metamodeling Environment
GME Metamodeling Paradigm
XTEAM ADL Metamodel
GME Domain-Specific Modeling Environment
XTEAM ADL
XTEAM Architecture Models
XTEAM Model Interpreter Framework
XTEAM Simulation Generators
adevs Simulation Engine
Application Simulations
xADL Structures and Types
Finite State Processes
ADL Extensions
Application Architectures
Energy Consumption Analysis
Reliability Analysis
End-to-end Latency Analysis
Memory Usage Analysis
Scenario-driven Analysis Results

The XTEAM Model Interpreter Framework

- Implements a mapping from the XTEAM domain-independent component model to a discrete event simulation model

- Employs the Strategy pattern to enable an architect to implement domain-specific extensions

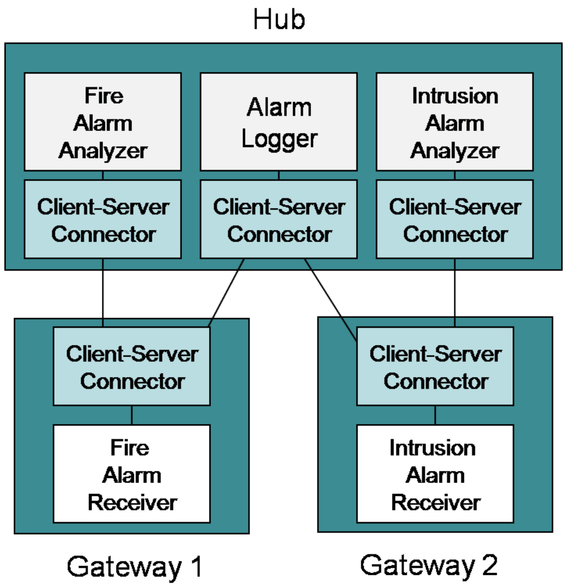
- Each Concrete Strategy generates code to realize a particular analytic theory
  - Invoked at specific times during the interpretation process


Context
XTEAM_Interpreter_Framework
generate()
1
*
Strategy
XTEAM_Simulation_Generator
on_input()
on_output()
on_task_begin()
on_task_end()
XTEAM_Latency_Sim_Generator
XTEAM_Memory_Sim_Generator
Concrete Strategies
XTEAM_Energy_Sim_Generator
XTEAM_Reliability_Sim_Generator

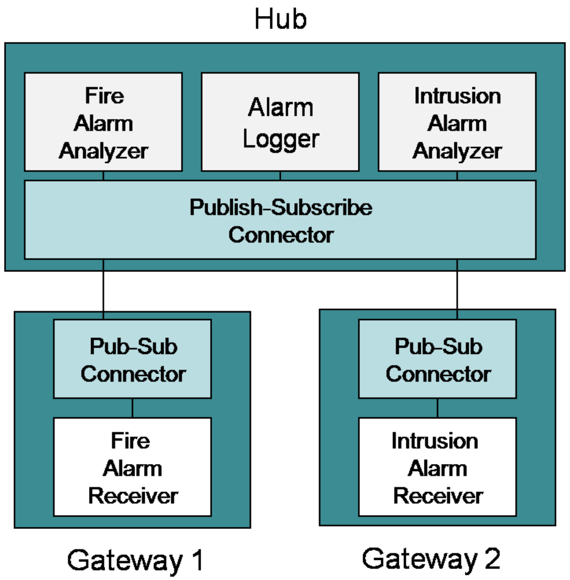
- Generated code calculates and records analysis results
  - Invoked when a component sends or receives data, calls an interface, starts or completes a task, *etc.*

Evaluation

- Evaluated using the MIDAS system, a family of sensor network applications
  - Runs on Prism-MW, a lightweig architectural middleware platf
- Two candidate architectures were analyzed: client-server and publish-subscribe
- XTEAM was used to determine the most energy efficient architectural style
  - Predictions of system properties made by XTEAM were compared with measured values taken from the executing system



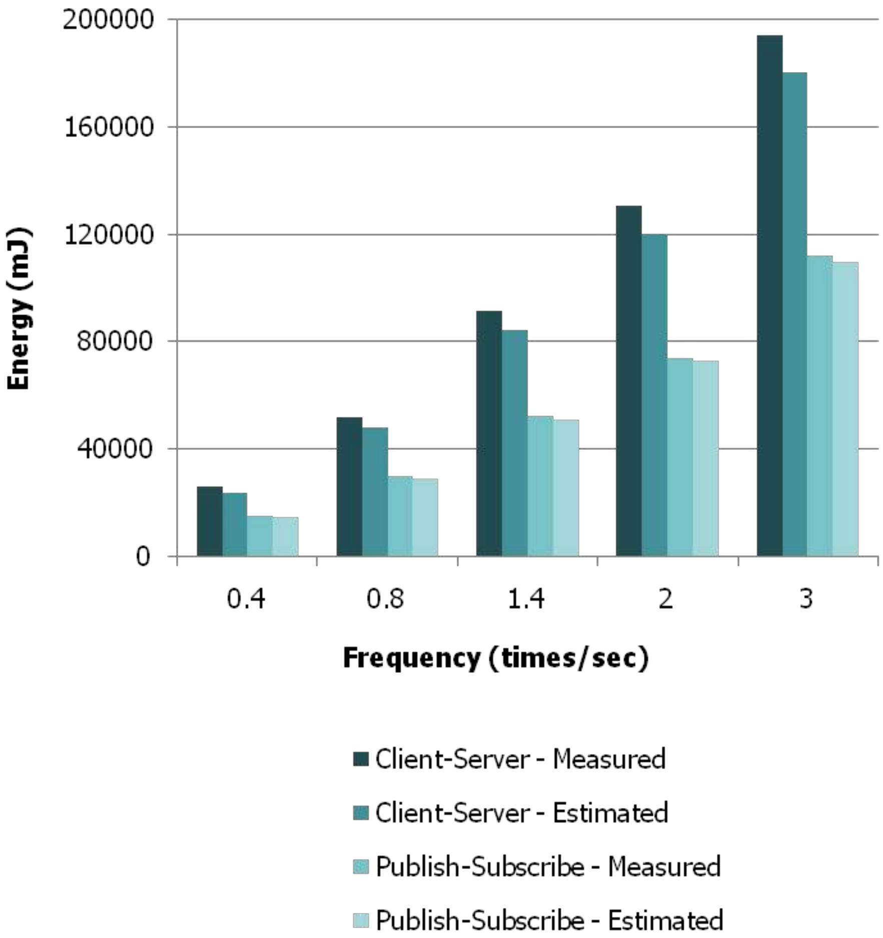
Client-server architecture



Publish-subscribe architecture

Verification

- The predicted energy consumption fell within 10% of the measured energy consumption in all scenarios
- The pub-sub style was more energy-efficient
  - Requires fewer events to be sent over the wireless network
  - The energy overhead due to processing subscription requests and retrieving subscriber lists is small
- The margin of error was sufficiently small that it led to the correct choice of architectural style



Conclusions

- Model interpreter frameworks enable MDE toolchains with *built-in* analysis capabilities along with metamodeling and domain-specific extensibility
  - Must make several important assumptions about the models to which they are applied
  - Must fulfill a set of design requirements

- For more information
  - Visit the XTEAM homepage:

    http://www-scf.usc.edu/~gedwards/xteam.html
  - Email George Edwards:

    gedwards@usc.edu